

Baze de date

- Proiect Tema-

Aplicatie Social Media

Damian Gabriel-Mihai
1308B

Cadru coordonator: Cristian-Nicolae Buțincu

Github: <https://github.com/GabrielDamian/social-media-app>

Descrierea proiectului:

Proiectul isi propune sa creeze o aplicatie web demo de tipul social media care sa fie capabila sa isi manipuleze totodata si baza de date.

Aplicatia este impartita in doua regiuni bine definite:

‘User section’ - Contine route care in mod normal ar fi disponibile utilizatorilor obisnuiti. De aici, acestia isi pot modifica date ale profilului propriu dar si adauga noi postari in sectiunea de ‘News Feed’ (postari vizibile de toti ceilalti useri din aplicatie).

‘Admin Section’ - Contine route destinate utilizatorilor de tip admin, prin intermediul carora pot modifica (CRUD type) datele din baza de date.

Accesul in aplicatie se face prin signup, routele prezentate mai sus(atat din ‘user section’ cat si ‘admin section’ fiind indisponibile utilizatorilor neinregistrati inca).

Fiecare utilizator nou creat are niste attribute (date personale), care poti fi vizualizate si editate din sectiunea ‘My Profile’. Acestea includ o descriere de tip text dar si un status (acest status nu reprezinta rolul utilizatorului in aplicatie, ci este doar alt mod de descriere al userului, un utilizator putand sa aiba mai multe statusuri simultan, alese dintr-o lista bine definita).

Tehnologii utilizate:

Front-end-ul aplicatie este bazat pe libraria React(JavaScript), folosind CSS pentru modelare GUI.

Paginile aplicatie au fost impartite in 8 route:

Inregistrare users:

/login

/signup

User Section:

/myprofile

/news-feed

Admin Section:

/table-users
/table-desc
/table-status
/table-posts

Pentru backend s-a folosit Node.js, serverul fiind bazat pe Express.js .
Pe parcursul aplicatiei, s-au folosit mai multe module node.js, dintre care cele mai importante fiind 'mysql' pentru a comunica cu baza de date din MySQLWorkBench, 'bcryptjs' pentru a codificata parola fiecarui user inainte de a fi stocata in tabela aferenta, dar si 'jsonwebtoken' pentru a stoca date despre userul curent in LocalStorage (aplicatia persista la refresh).

Structura si inter-relationarea tabelelor:
Aplicatia este baza pe 4 tabele dupa cum urmeaza:

Tabela USERS:

- retine date despre fiecare user nou inregistrat in aplicatie
- password_token este un string obtinut prin codificarea parole userului in combinatie cu un 'secret key', decodificarea facandu-se prin bcryptjs ori de cate ori acelasi user isi face login din nou.

Field	Type	Null	Key
ID	varchar(255)	NO	PRI
username	varchar(255)	NO	
password_token	varchar(255)	NO	

Tabela STATUS:

- fiecarui user ii pot fi atribuite mai multe statusuri dintr-o lista bine definita, relatie impusa prin check;
- nu poti exista statusuri care sa nu apartina vreunui user (foreign key cu user_id din users)

Field	Type	Null	Key
user_id	varchar(255)	NO	MUL
status_value	varchar(255)	YES	

Tabela DESCRIPTION:

- fiecarui user ii poate fi atribuita o singura descriere la un moment dat; (foreign key cu user_id din users)

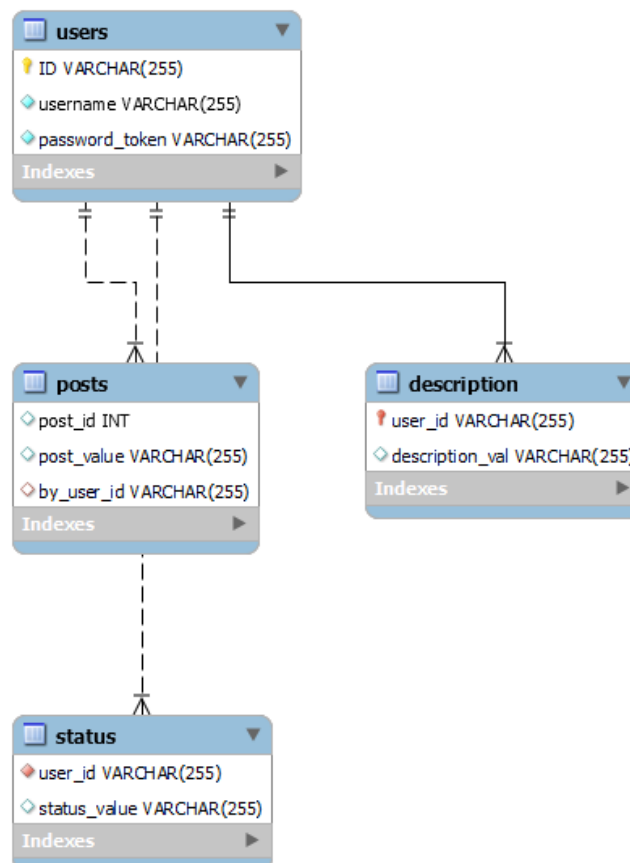
Field	Type	Null	Key
user_id	varchar(255)	NO	PRI
description_val	varchar(255)	YES	

Tabela POSTS:

- o postare se defineste prin 3 campuri: id, continut si user;
- continutul este folosit in cadrul aplicatiei ca un link catre o imagine ce v-a fi afisata in sectiunea de news-feed.
- pot exista postari 2 inregistrari in tabela cu acelasi id, in acest caz intelegandu-se ca cei de users sunt implicati in crearea aceleasi postari, acestia fiind de asemenea afisati in sectiune de news-feed in dreptul postarii. (aplicatia in acest moment permite doar crearea de postari in care sunt implicati 2 users, link-ul imaginii dintre cele doua inregistrati cu acelasi id, fiind setat acelasi, in mod automat).

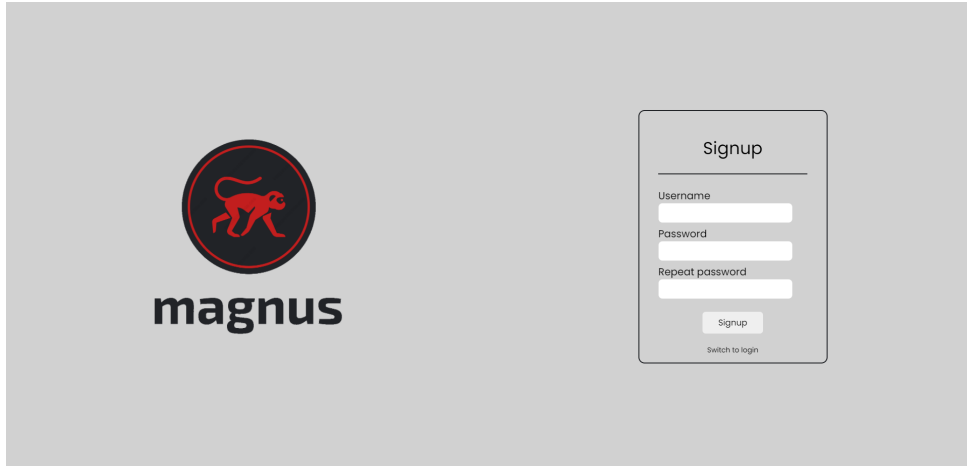
Field	Type	Null	Key
post_id	int	YES	
post_value	varchar(255)	YES	
by_user_id	int	YES	

Schema db:



Screenshots app:

Login/Signup:



Signup

Username

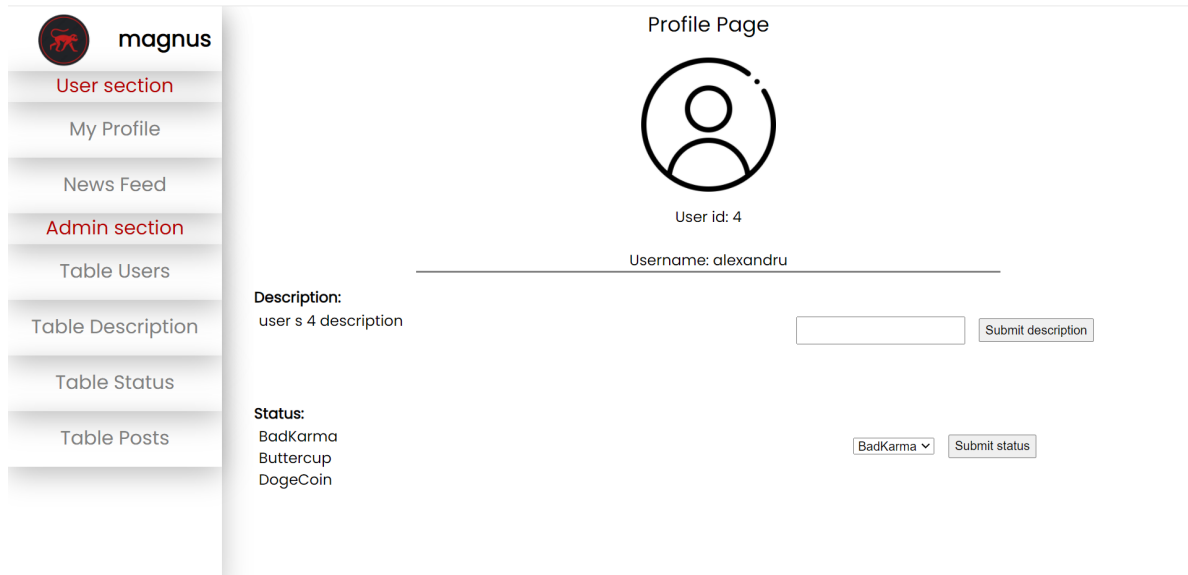
Password

Repeat password

Signup

[Switch to login](#)

Profile page:



magnus

User section

My Profile

News Feed

Admin section

Table Users

Table Description

Table Status

Table Posts

Profile Page

User id: 4

Username: alexandru

Description:

user s 4 description

Submit description


Status:

BadKarma
Buttercup
DogeCoin

BadKarma

Submit status

News Feed:

**magnus**

User section

My Profile

News Feed

Admin section

Table Users

Table Description

Table Status

Table Posts

News Feed

Post new:


Tag other:

damian

Post


Post id: 8

by: 4 and 2




Post id: 6

by: 3 and 6



Tabel:

**magnus**

User section

My Profile

News Feed

Admin section

Table Users

Table Description

Table Status

Table Posts

TABLE USERS

Current userID: 4

Log out

Add Item:

Username

Password

Add into table

Modify Item:

Select item ID:

1

Username

Password

Update into table

ID	Username	Password Token	Delete
1	damian	\$2a\$10\$Dfh4C3DkGNy/20/3B.3zoQYq96QzQbc7aCve2AW5ZkxfXBAQ0JtMu	<div></div>
2	gabriel	\$2a\$10\$iveYW7oP2Cloo.nK4WuPoutNkwzGj9YzomnvgakHGVt//DShEnJGm	<div></div>
3	mihai	\$2a\$10\$OmUW8ODVzG7VMuSE9S9EXeJxkryJ8TPKj4JJRYI/IRsijnp0MR.Ae	<div></div>
4	alexandru	\$2a\$10\$jJouKgzCIS5V2w0YiAobyE0N0s9pJX3JLWUzCBvLrpiq8yITsklaW	<div></div>
5	marian	\$2a\$10\$zT5zwmH/uJEGmtlTHPKIqu4qNctn0aU56jq3HfqCultO4M2ET3BJu	<div></div>
6	robert	\$2a\$10\$UKpb2KA2BgRldfY4IUfip.skbsHm36BbO3hs.OF6bhm32plOr2/lu	<div></div>

Exemple sql:

-verificare existenta user(folosita la signup)

```
con.query(`select username from users where username = '${username}'`, async (err, data)=> {
  if (err) {
    return res.status(400).send(err);
  }
})
```

-colectare date din tabela USERS

```
app.post('/api/fetch-table-users', async (req, res) => {  
  
  console.log('fetch-table');  
  con.query(`select * from users;`, async (err, data) => {  
    if (err || data.length == 0) {  
      return res.status(400).send(err);  
    }  
  })  
})
```

-inserare in tabela DESCRIPTION

```
con.query(`insert into description values(${user_id}, '${desc_val}')`, async (err, data) => {  
  
  if (err || data.length == 0) {  
    return res.status(400).send(err);  
  }  
})  
res.json({  
  'lines': data  
})  
});
```